Assignment 1: Ray Tracing

Handout date: 9/19/2017
Submission deadline: 10/15/2017, 23:59 EST
Demo date: 10/17/2017 2-3PM

This homework accounts for 17.5% of your final grade.

Goal of this exercise

In this exercise you will write a raytracer, and use it to render spheres and more complex triangulated surfaces.

Eigen

In all exercises you will need to do operations with vectors and matrices. To simplify the code, you will use `Eigen`. Have a look at the "Getting Started" page of `Eigen` as well as the Quick Reference page for a reference of the basic matrix operations supported.

Submission

1. Follow the link (to be sent by email) to create your repository in `https://github.com/NYUCG2017/assignment-1_USER` with starter code.
2. Follow further instruction to set up travis badge.
3. Modify the code following the assignment instructions
4. Add a readme in pdf or markdown format as a report of what you did containing a screenshot for each task
5. Push the code into the repository before deadline and make sure travis-ci build passes.

Questions

We advise all non-private questions be posted on `https://github.com/danielepanozzo/cg/issues` as reference for all students. For other questions, please email us or come to the office hours.

1 Mandatory Tasks

For each tasks below, add at least one image in the readme demonstrating the results. The code that you used for all tasks should be provided.

1.1 Ray Tracing Spheres

Modify the given program to support the rendering of multiple spheres in general position. Note that the provided code is only an example and the collision code that it uses is not general. You need to reimplement it from scratch. For this step, use simple Lambertian shading.

1.2 Shading

Extend your ray tracer to support ambient and specular lighting. Render a scene with multiple spheres with different colors and different material properties (one sphere should be purely diffuse, another one specular). Add a second light source to your scene.

1.3 Perspective Projection

Extend your ray tracer to support perspective projection, and re-render the scenes you created in the previous tasks, showing the difference between the two projections.

1.4 Ray Tracing Triangle Meshes

Extend your ray tracer to load meshes in off format. Load the two meshes provided in the *data* folder and add them to your scene. Render them with different colors. Suggestion: You can store your mesh with two `Eigen` arrays V and F. V is a float array (dimension #V $\times$ 3 where #V is the number of vertices) that contains the positions of the vertices of the mesh, where the i-th row of V contains the coordinates of the i-th vertex. F is an integer array (dimension #faces $\times$ 3 where #F is the number of faces) which contains the descriptions of the triangles in the mesh. The i-th row of F will contain the indices of the vertices in V that form the i-th face, sorted counter-clockwise.

1.5 Shadows

Add shadows to the previous scene.

1.6 Reflections on the floor

Add a mirror that reflects all objects in the scene. To simplify this task, you can simply transform the entire "floor" of the scene into a mirror.

Optional Tasks.

These tasks are optional. Each one of these tasks is worth 1.5% of the final grade. The optional points are added to the points of the other exercises, but the total sum of points that you gain with exercises cannot be more than 80%.

1.7 Parallelization

Each pixel of the scene can be rendered independently. Integrate Intel TBB in your project and use the *parallel for* to distribute the computation over all the available cores.

1.8 Animation

Create a simple animation by rendering multiple frames while moving the position of the camera and of the light source.